

Fig. 1

***SizingPathGeneration(set1)***

foreach leaf cell **cella** in **set1**

find all the half-operators in **cella** that are going to start a path, put in **seta**

foreach half-operator **hoa** in **seta**

**lsta** is an empty list of half-operators

RecursivePaths(**lsta**, **hoa**)

***RecursivePaths(lst1, ho1)***

If (**lst1** contains **ho1**) make sizing-path from **lst1**, then return // found cycle

If (**ho1** drives an observable point) make sizing-path from **lst1** and **ho1**, then return

foreach **hoa** that is driven by **ho1** and has opposite driving direction of **ho1**

RecursivePaths(**lst1+ho1**, **hoa**)

**Fig. 2**

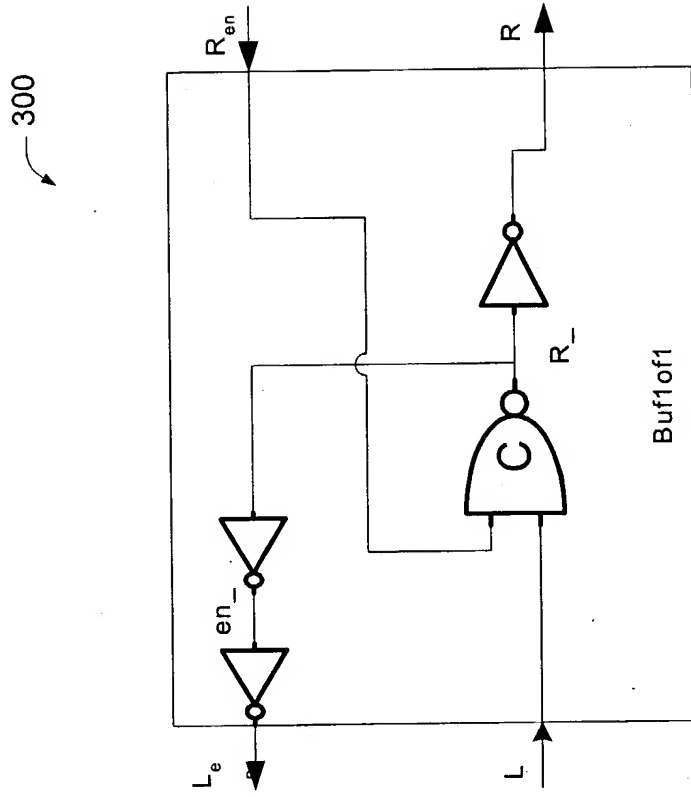


Fig. 3

***CatPathGeneration(set1)***

**lsta** is sorted list of mid-level cells in **set1**, from low to high (e.g. from mid to top levels)

foreach **cella** in **lsta**

    // N.B. an nonobservable sucells is a subcell which contains

    // a nonobservable point

    pop up paths from nonobservable subcells of **cella** to **cella**

    find all the paths in **cella** that are going to start a cat-path, put in **seta**

    foreach **patha** in **seta**

**lsta** is an empty list of paths

        RecursiveCatPaths(**lsta**, **patha**)

***RecursiveCatPaths(lst1, path1)***

if (**lst1** contains **path1**) make cat-path from **lst1**, then return   // found cycle

if (last half-operator of **path1** drives an observable point)

    make cat-path from **lst1** and **path1**, then return

foreach **patha** that is driven by **path1**

RecursiveCatPaths(**lst1+path1**, **patha**)

**Fig. 4**

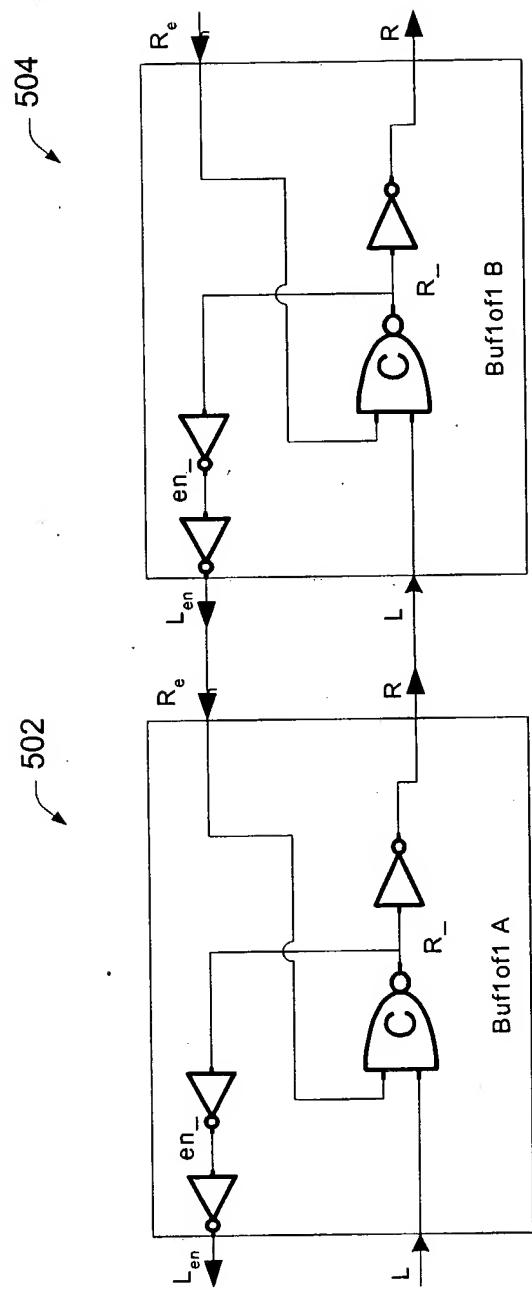
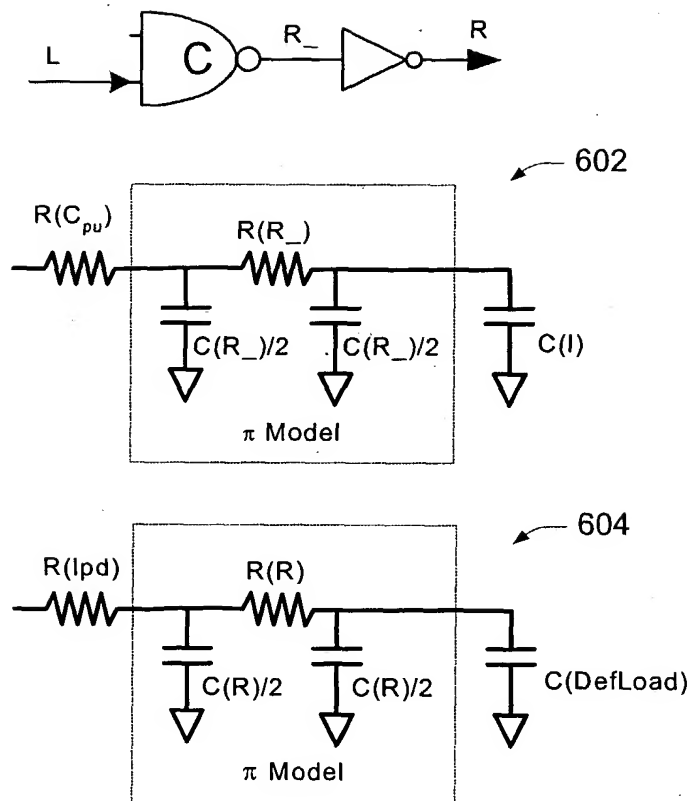


Fig. 5



**Fig. 6**

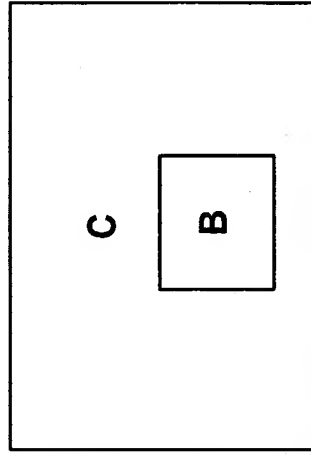
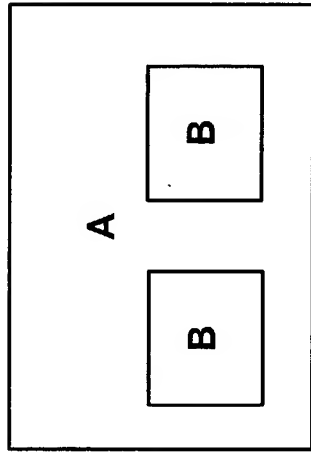
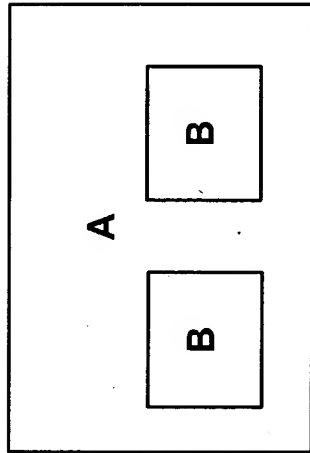


Fig. 7

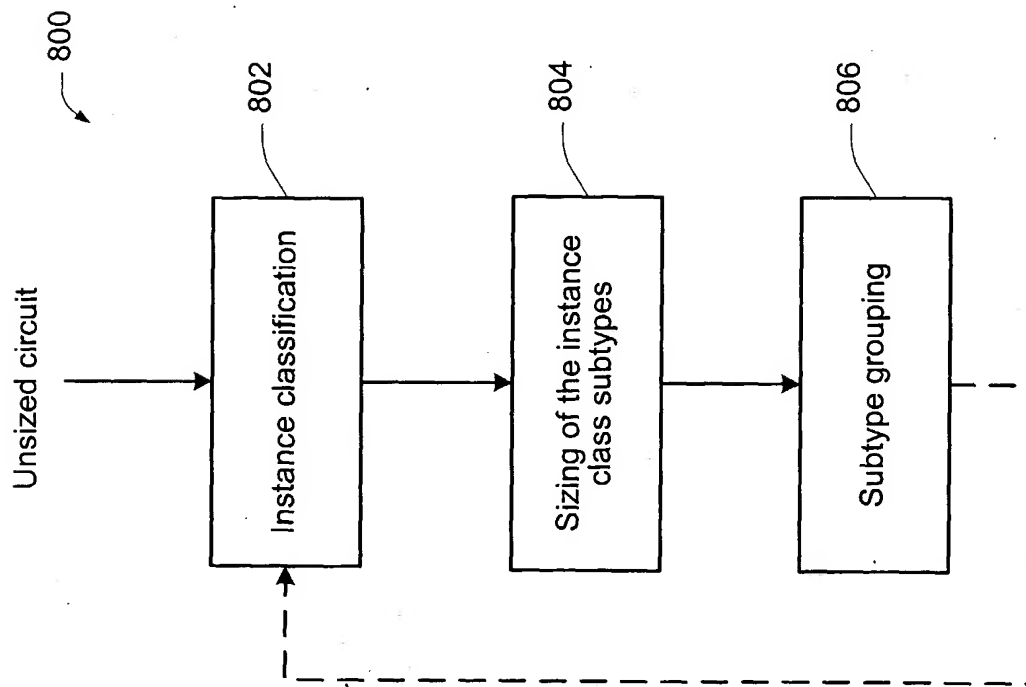


Fig. 8

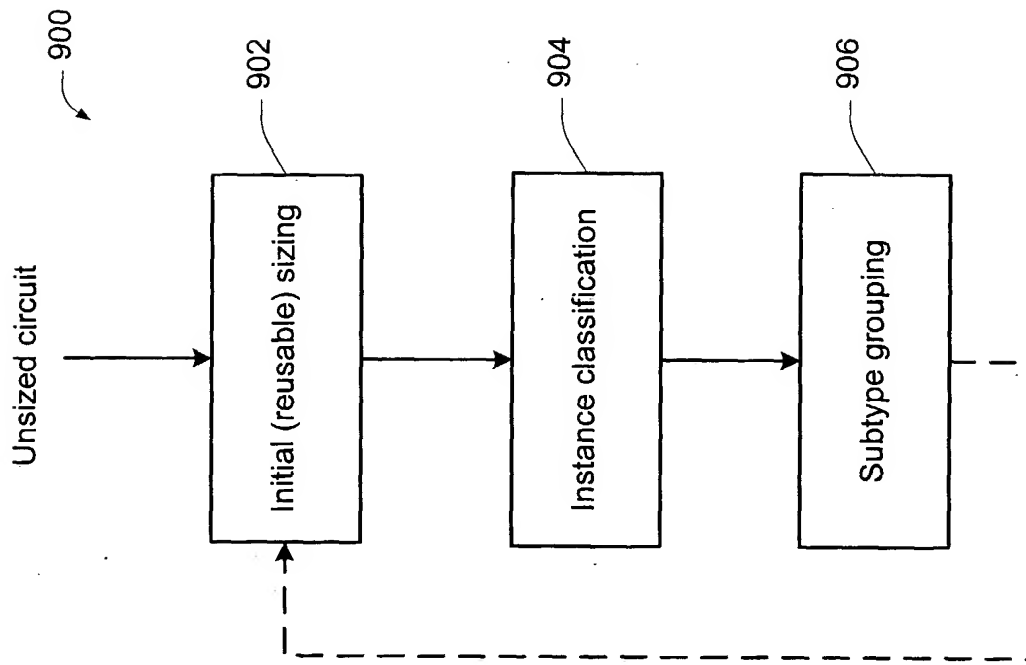


Fig. 9